Sandra Cardenas
June 26, 2025

Assignment Name: CS 470 Final Reflection

Presentation Video Link: https://youtu.be/clhEFtZp_QM?si=XeFLZ-LZpWfeF1CA

Experiences and Strengths

Throughout CS 470, I developed hands-on skills that strengthened my understanding of cloud-native web development. This course not only enhanced my technical abilities but also gave me a real-world perspective on what it's like to architect scalable and efficient applications using modern cloud services. I became proficient in Docker and Docker Compose, containerizing applications for consistency and ease of deployment. I also learned how to deploy static frontend apps using AWS S3, and how to build and integrate backend APIs using AWS Lambda and DynamoDB. These are highly in-demand skills that make me a more marketable candidate as I pursue roles in software and cloud development.

One of my biggest strengths as a developer is adaptability. Even when faced with tools I hadn't used before, such as LoopBack or IAM policy configuration in AWS, I was able to troubleshoot, experiment, and ultimately implement solutions that worked. I'm also detail-oriented, and I've learned to work through full-stack issues from the frontend to backend, including error handling, cross-origin permissions, and integrating cloud APIs with Angular services.

With the knowledge and experience gained in CS 470, I feel prepared to take on roles such as Cloud Developer, Junior DevOps Engineer, or Full Stack Developer at companies seeking modern, cloud-first solutions. I can contribute to projects that rely on scalable infrastructure and API-driven architecture, especially those that benefit from serverless technologies.

Planning for Growth

Understanding how to plan for the future scalability of a web application is one of the most valuable lessons I've taken from this course. As applications grow, the ability to predict cost, handle traffic surges, and ensure high availability becomes critical. CS 470 introduced me to serverless and microservice-based design, which are game changers in achieving these goals.

Serverless architecture offers inherent elasticity. Functions only run when triggered, scaling automatically in response to user demand. This makes it ideal for unpredictable workloads or startups where traffic can spike at any moment. AWS Lambda and API Gateway were excellent tools for building lightweight, cost-efficient APIs. In the future, I could extend my application by breaking features into smaller Lambda functions that handle specific tasks, allowing for more granular control, isolation, and performance tuning.

When comparing serverless vs. containers, cost predictability becomes a key factor. Serverless is more cost-effective for low-to-medium workloads because you pay only for usage. Containers (like those run with ECS or Kubernetes) provide more control over runtime environments and are better suited for long-running or resource-heavy tasks, but they often incur steady costs whether the app is being used or not. For most use cases, serverless is more efficient to start with, while container-based microservices might be considered during larger-scale operations with continuous workloads.

Error handling and monitoring are also important as systems scale. In serverless architecture, I can set up CloudWatch logging and custom error responses within the Lambda and API Gateway settings. This means my app can provide useful feedback when failures occur without needing manual intervention.

Elasticity and the pay-for-use model allow organizations to remain agile. Rather than investing in infrastructure upfront, I can let AWS handle scaling, storage, and provisioning, which is especially beneficial for rapid development cycles. This means less time managing infrastructure and more time focusing on user experience and application logic.

As my application evolves, these cloud-native principles will guide how I build: prioritize scalability, isolate concerns through microservices, and keep costs efficient through event-driven and serverless design. This course gave me a solid foundation to confidently plan for both the technical and financial growth of web applications in the cloud.